编译原理
实验四：目标代码生成

规格严格，功夫到家

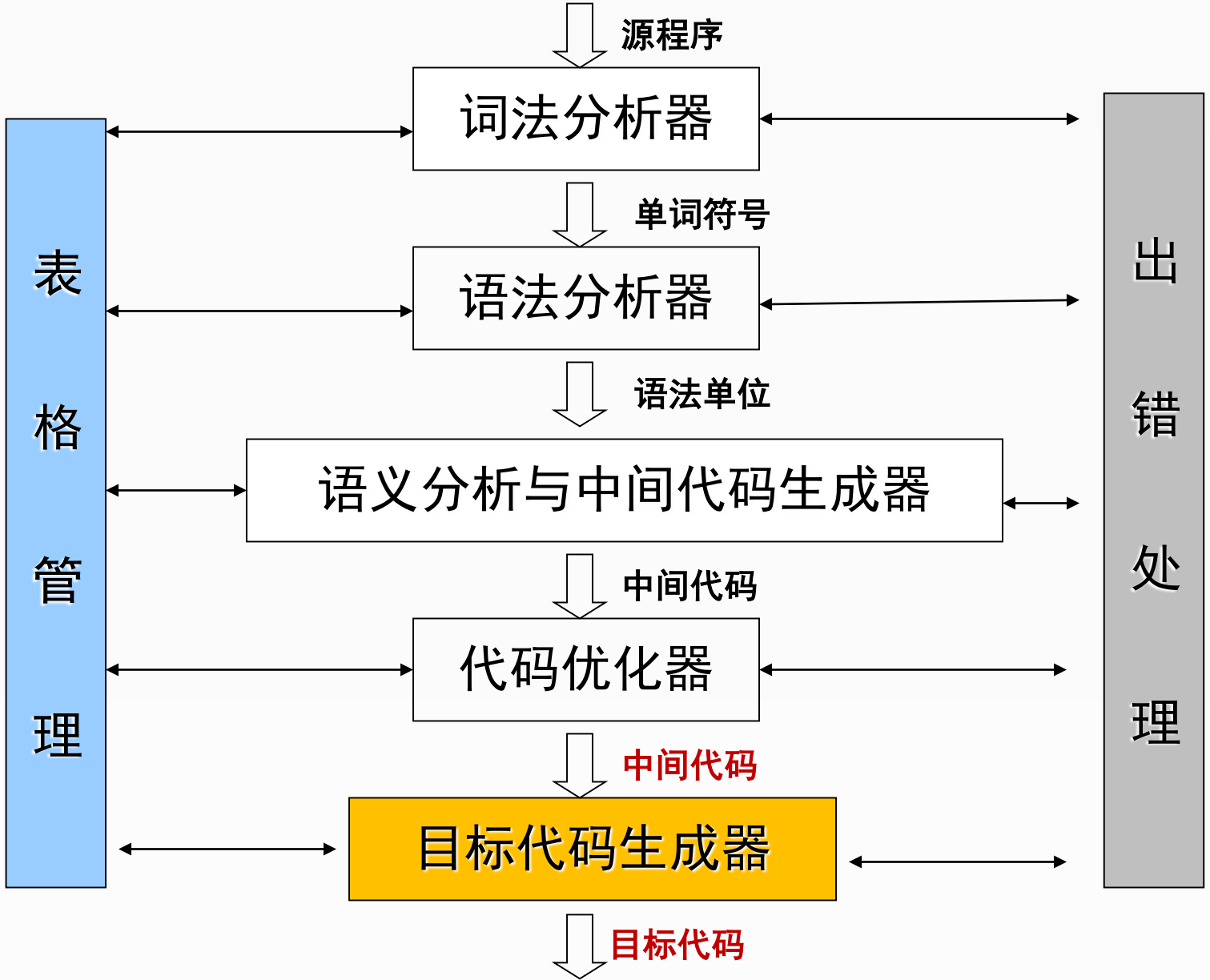
编译程序的总体结构

✔实验一：

✔实验二：

✔实验三：

✔实验四：



实验目的

1. 加深对编译器总体结构的理解与掌握；

2. 掌握常见的RISC-V指令的使用方法；

3. 理解并掌握目标代码生成算法和寄存器选择算法。

实验学时数：2学时

实验内容

1. 将实验三生成的中间代码转换为目标代码（RISC-V指令）；
2. 使用RARS运行生成的目标代码，验证结果的正确性。

```
data/in
├── coding_map.csv        # 码点文件
├── grammar.txt           # 语法文件
├── input_code.txt        # 输入代码
├── LR1_table.csv         # （可选）第三方工具生成的 IR 分析表
└── reg-alloc.txt         # （可选）用于测试寄存器分配的样例
```

选做

```
data/out
├── assembly_language.asm    # 汇编代码
├── intermediate_code.txt    # 中间表示
├── ir_emulate_result.txt    # 中间表示的模拟执行的结果
├── parser_list.txt          # 规约过程的产生式列表
├── new_symbol_table.txt     # 语义分析后的符号表
├── old_symbol_table.txt     # 语义分析前的符号表
└── token.txt                # 词法单元流
```

实验四输出

- 实验四不检查out/assembly_language.asm和std/assembly_language.asm是否一致，而是使用rars运行汇编指令后，查看寄存器a0中的值是否正确。
- rars使用参考指导书

rars下载链接：https://gitee.com/hitsz-cslab/Compiler/releases/tag/2022F.0.1

目标代码生成算法（RISC-V）

对每个形如（op，result,lhs,rhs）的三地址语句

1. 使用寄存器选择算法为result选择寄存器。
2. 如果左操作数lhs已经在寄存器中，使用当前寄存器，右操作数rhs同理。
3. 否则，如果左操作数lhs是变量且在内存中，使用寄存器选择算法为它选择一个寄存器，将内存中的变量值加载到寄存器中，右操作数rhs同理。
4. 生成汇编指令。

注：以上算法仅供参考，可以有不同实现。

寄存器选择算法（RISC-V）

1. 如果有空闲寄存器，选择空闲寄存器；
2. 否则，夺取不再使用的变量所占的寄存器；
3. 如果所有寄存器中的变量后面都要使用，自行设计算法从被占用的寄存器中夺取一个，并将被夺取寄存器的变量存回内存。（此步骤可选，不加分）

备注：

- 在代码生成时，约定使用RISC-V临时寄存器：t0-t6
- 同时使用 a0，亦即 x10，存放程序的返回值
- 使用input_code.txt样例只需要实现寄存器选择算法的1，2两个步骤
- 使用data/in/reg-alloc.txt样例需要实现寄存器选择算法的1，2，3三个步骤
- 以上算法仅供参考，可以有不同实现

目标代码生成举例

三地址码：（ADD,$1,a,b），假设a在t0、b在t1寄存器中

- 为$1选择空闲寄存器t2；

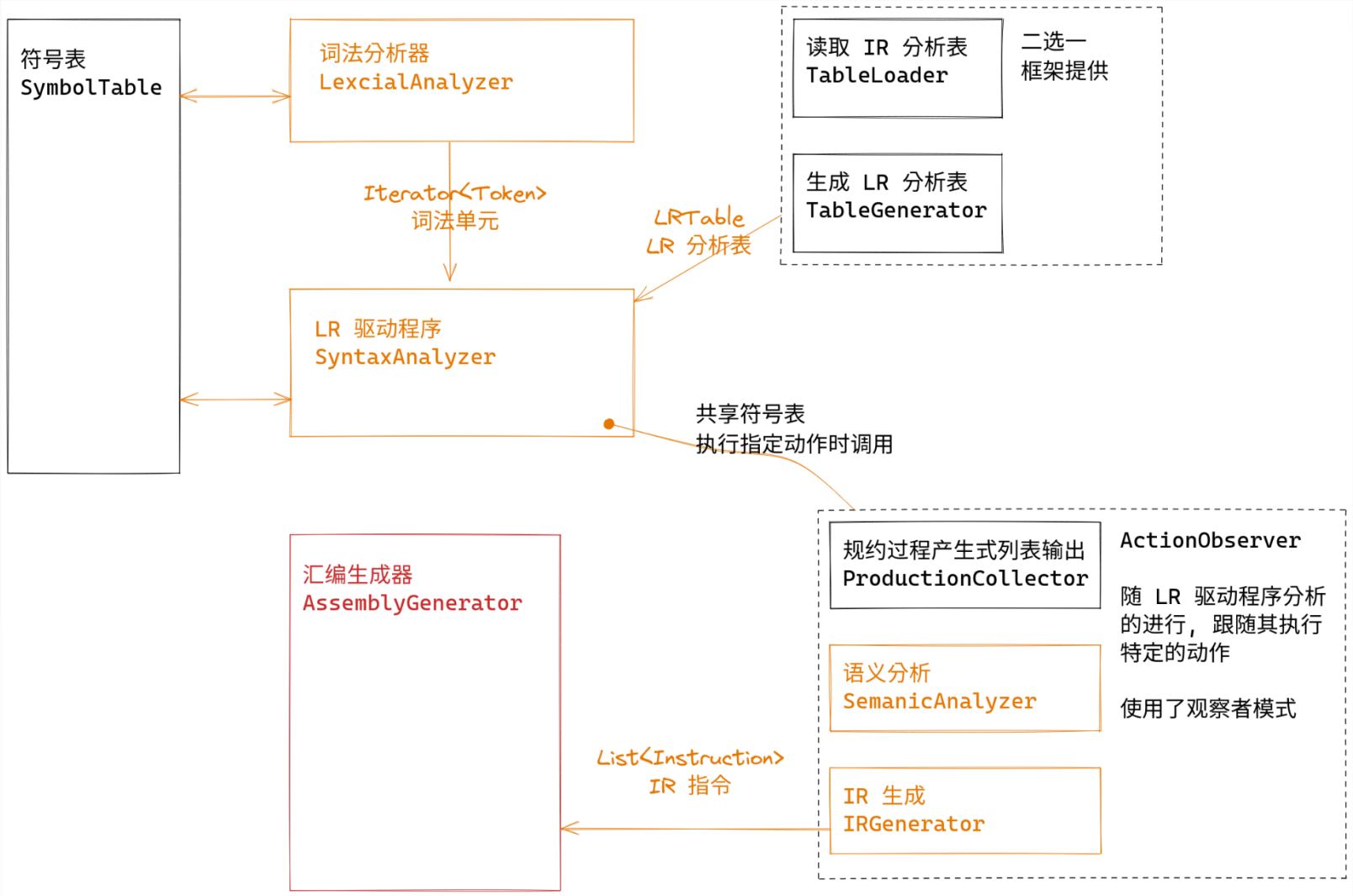
目标代码：add t2, t0, t1

三地址码：(ADD, $2, b, 3)，假设b在t3寄存器

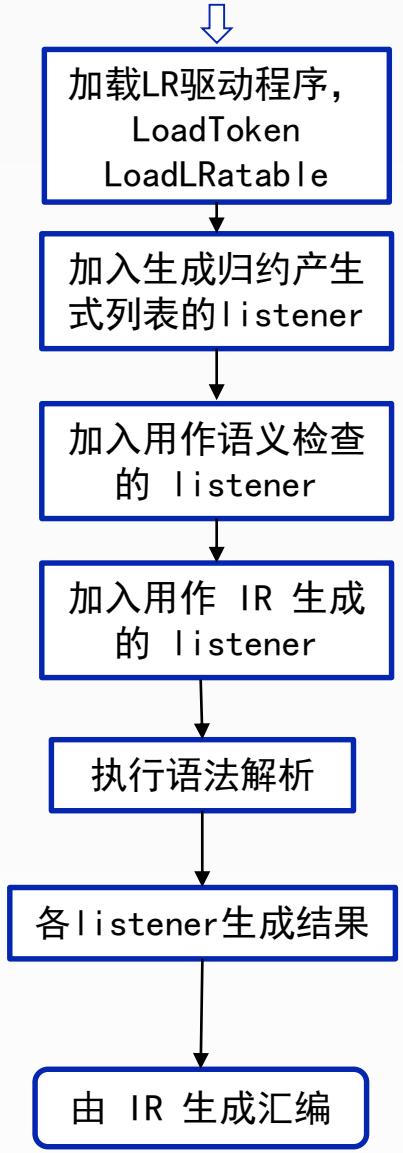
- 为$2选择空闲寄存器t4；

目标代码：addi t4, t3, 3

编译器程序整体框架图


符号表
SymbolTable
词法分析器
LexcialAnalyzer
读取 IR 分析表
TableLoader
二选一
框架提供
生成 LR 分析表
TableGenerator
Iterator<Token>
词法单元
LRTable
LR 分析表
LR 驱动程序
SyntaxAnalyzer
共享符号表
执行指定动作时调用
汇编生成器
AssemblyGenerator
规约过程产生式列表输出
ProductionCollector
ActionObserver
随 LR 驱动程序分析的进行，跟随其执行特定的动作
语义分析
SemanicAnalyzer
使用了观察者模式
List<Instruction>
IR 指令
IR 生成
IRGenerator

主程序流程说明




```
final var parser = new SyntaxAnalyzer(symbolTable);
(lrTparser.loadTokens(tokens);
parser.loadLRTableable);

final var productionCollector = new
ProductionCollector(GrammarInfo.getBeginProduction());
parser.registerObserver(productionCollector);

final var semanticAnalyzer = new SemanticAnalyzer();
parser.registerObserver(semanticAnalyzer);

final var irGenerator = new IRGenerator();
parser.registerObserver(irGenerator);

parser.run(); //语法分析器驱动程序

productionCollector.dumpToFile(FilePathConfig.PARSER_PATH);  //输出产生式列表
symbolTable.dumpTable(FilePathConfig.SYMBOL_TABLE_PATH);     //输出符号表
final var instructions = irGenerator.getIR();
irGenerator.dumpIR(FilePathConfig.INTERMEDIATE_CODE_PATH);   //输出中间代码

final var asmGenerator = new AssemblyGenerator();
asmGenerator.loadIR(instructions);
asmGenerator.run();
asmGenerator.dump(FilePathConfig.ASSEMBLY_LANGUAGE_PATH);
```

语法分析、语义分析与中间代码生成的三个观察者

实验步骤

1. 加载实验三生成的中间代码，视情况做预处理（预处理思路参考指导书）；
2. 实现寄存器选择算法；
3. 实现目标代码生成算法；
4. 输出生成的目标代码到指定文件中；
5. 使用scripts中的check-result.py检查

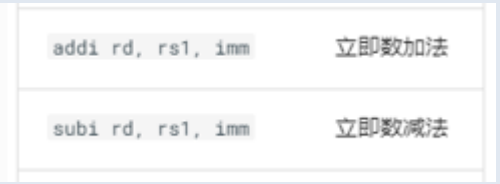
实验四结果时，先要修改文件中的rars_path。

```
# 需要改为你自己的 rars.jar 路径
rars_path = "/home/test/rars.jar"
```

预处理举例

中间代码：(ADD, $2, 3, b)

RISC-V 指令：



预处理后的中间代码：(ADD, $2, b, 3)

欢迎同学进群了解竞赛信息！！！

群名称:HITSZ-编译器竞赛交流群
群　号:580679631

全国大学生计算机系统能力大赛——编译系统设计赛
https://os.educg.net/#/index?TYPE=COM

同学们，
独立开始实验